H10019JDP
Customer No. 01333

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:
Edward Neil Chapman

PRINTING SYSTEM AND METHOD FOR CUSTOMIZATION OF A PRINT JOB

Serial No. 09/731,503

Filed 06 December 2000

Group Art Unit: 2626

Examiner: Michael Burleson

I hereby certify that this correspondence is being deposited today with the United States Postal Service as first class mail in an envelope addressed to Commissioner For Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Debra Nowacki

Debra Nowacki

February 16, 2006

Date

DECLARATION UNDER 37 C.F.R. 1.131

I, Edward Chapman, do hereby declare that:

1. I am the inventor of the above-identified patent application.

2. Prior to May 17, 2000, I completed the invention as described and claimed in the subject application in this country, as evidenced by the following.

a. I, prior to May 17, 2000, having earlier conceived the ideas represented in Claims 1-20, attached hereto as Exhibit A, did reduce such ideas to practice as evidenced by the figures and software code attached hereto as Exhibits B-F.

b. Figure 1 of Exhibit B illustrates a printing workflow, which was improved upon by the present invention.

c. Figure 2 of Exhibit B illustrates the improved printing workflow according to the present invention as recited by the claims shown in Exhibit A.

d. The printing workflow illustrated by Figure 2 of Exhibit B and described below was conceived of and reduced to practice prior to May 17, 2000.

e. According to the invention as illustrated by Figure 2 of Exhibit B, a user editing a document with an application ("APP"), such as Microsoft Word, submits such document for printing, e.g., by clicking a print button.

f. After submitting the document for printing, as described at 2e above, a user interface ("UI") is presented to the user that allows the user to select features, such as stapling, supported by the printer ("marking engine" or "ME").

g. After selecting features to be applied to the printing of the document, as described at 2f above, a print driver converter ("PDC") converts the application file into a page description language ("PDL") file while inserting a description of the features selected by the user into the PDL file.

h. The functionality added to the PDC, prior to May 17, 2000, allows the user to choose the features on the user interface ("UI") and inserts commands into the PDL file used to execute the appropriate plugins ("plugins") that implement the selected feature(s).

i. An example of inserting a command into the PDL file by the PDC, as described in 2h, above, is for the PDC to insert "%KDKCustom: on -dstore" into the PDL file.

j. The component DL in Figure 2 of Exhibit B illustrates a downloader software component that performs the same functions as the PDC, but takes a PDL file as input and, consequently, does not perform a PDL conversion.

k. The changes to the PDC and DL described at 2g, 2h, 2i, and 2j above were simple in nature and source code showing such changes has not been provided.

l. The raster image processor ("RIP") illustrated in Figure 2 of Exhibit B converts the input PDL into data representing dots to be printed by the ("ME").

m. The invention modified the RIP, described at 2l above, to (1) call the customization detector ("CD") with the commands provided by the DL or PDC; (2) fill in a list of functions to be called at the start of a job, with each image in the job, and at the end of the job; and (3) call these functions ("plugins") at the correct times to implement the features selected by the user.

n. The source code implementing the CD and a plugin, which are called by the RIP, as described at 2m, above, are attached hereto as Exhibits C and D, respectively.

o. The CD, illustrated in Figure 2 of Exhibit B, informs the RIP of which plugins and arguments for the plugins may be applied in the function CheckCustom(), which calls the function DeterminDLs(), both of which

are shown in the source code CheckCustom.c attached as Exhibit C.

p. A plugin, illustrated in plurality in Figure 2 of Exhibit B as "plugins", provides three functions, one called at the start of a job, one with each image, and the last at the end of a job.

q. A plugin providing a RIP to store feature that met the definition described at 2p above, was implemented prior to May 17, 2000.

r. The source code for the RIP to store feature described at 2q above is the file CustomWriteTiff.c, attached hereto as Exhibit D, wherein the Custom() function performs the RIP to store and calls the WriteTiff() function to create TIFF files in WriteTiff.c attached hereto as Exhibit E.

s. The header file for the RIP to store function, described at 2q and 2r above, is attached hereto as Exhibit F.

t. Although the software code shown in Exhibit C-F include revisions described in the software code that were made subsequent to May 17, 2000, such software code, as it existed prior to May 17, 2000 without such revisions, was successfully executed prior to May 17, 2000 to perform one or more processes described by Claims 1, 2, 3, 4, 5, 6, 7, and 8 shown in Exhibit A as evidenced by:

    (i) the PDC, RIP, CD, and plugins described above, as pertinent to Claim 1;

    (ii) the RIP, plugins, and ME described above, as pertinent to Claim 2;

    (iii) the DL software component described above, as pertinent to Claim 3;

    (iv) the PDC and the detector program ("DET") illustrated with Figure 2 of Exhibit B, as pertinent to Claim 4, such that the DET chooses to route a file to either the DL component or the PDC component;

    (v) the CD component described above, as pertinent to Claim 5, such that the CD component retrieves plugins from a database;

    (vi) the PDC, DL, RIP, CD, plugins, and ME described above, as pertinent to Claim 6;

    (vii) the PDC described above, as pertinent to Claim 7; and

    (viii) the CD component described above, as pertinent to Claim 8,

such that the CD retrieves plugins from a database.

u. The software code shown in Exhibits C-F, as it existed prior to May 17, 2000, was successfully executed in one or more systems according to Claims 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, and 20, shown in Exhibit A, prior to May 17, 2000, as evidenced by statements 2t(i)-(viii), above.

v. The software code shown as Exhibits C-F, as it existed prior to May 17, 2000 and when executed as described in statements 2t and 2u, above, worked for its intended purpose.

w. I further declare under penalty of perjury pursuant to the laws of the United States of America that the foregoing is true and correct and that this declaration was executed by me on Feb 9, 2006 at Rochester, New York.

Edward Neil Chapman
Edward Neil Chapman

Feb 9, 2006
Date

42 Bending Creek Rd.
Street

Rochester NY 14624
City, State, Zip

EXHIBIT A

1. A method of customizing a print job, the method comprising the steps of:

receiving an input of an application file;

selecting a preferential document-processing feature from a group of document- processing features for a print job; and

applying a plug-in module, for supporting the preferential document-processing feature, to the application file.

2. The method according to claim 1 further comprising the step of printing at least a portion of the application file using the plug-in module for the print job.

3. The method according to claim 1 wherein the application file comprises a page description language file selected from the group consisting of a portable document format (PDF), printer control language (PCL), and a PostScript file.

4. The method according to claim 1 further comprising the step of:

determining whether or not the application file represents a page description language file;

converting the received application file into a page description language file if the received application file does not represent a page description file.

5. The method according to claim 1 wherein the selecting step comprises:

accessing a plug-in module database to retrieve the selected plug-in module.

6. A method of customizing a print job, the method comprising the steps of:

receiving an input of an application file;

converting the application file into a page description language file if the application file is in a format distinct from the page description language file format;

associating a preferential document-processing feature with the page description language file;

selecting a plug-in module associated with the preferential document-processing feature for a print job; and

printing the page description language file using the selected plug-in module for a print job.

7. The method according to claim 6 wherein the page description language file is in a form selected from the group consisting of a portable document format (PDF), printer control language (PCL), and a PostScript file.

8. The method according to claim 6 wherein the selecting step comprises:

accessing a plug-in database to retrieve the selected plug-in module.

9. A system for customizing a print job, the system comprising:

a detector for receiving an input of an application file and determining whether the application file represents a page description language file;

a user interface for selecting a preferential document-processing feature
from a group of document-processing features; and

a printer for applying a plug-in module, associated with the preferential document-processing features, to the application file.

10. The system according to claim 9 where the printer includes a bitmap printing module for printing the application file.

11. The system according to claim 9 wherein the application file comprises a page description language file selected from the group consisting of a portable document format (PDF), printer control language (PCL), and a PostScript file.

12. The system according to claim 9 further comprising:

a converter for converting the application file to a page description language file if the application file does not represent a page description language file.

13. The system according to claim 9 wherein the printer includes a customization detector, a plug-in selector, and a plug-in database; the customization detector configured to detect whether customization data is associated with the application file, the plug-in selector in communication with the customization detector and the plug-in database for selecting an active plug-in module based on the customization data.

14. A system of customizing a print job, the system comprising:

a detector for receiving an input of an application file and determining whether the application file represents a page description language file;

a data augmenter for associating a preferential document-processing feature with the application file; and

a plug-in selector for selecting a plug-in module for supporting the document-processing feature.

15. The system according to claim 14 comprising:

a printer for printing the application file using the selected plug-in module.

16. The system according to claim 14 wherein the application file comprises a page description language file selected from the group consisting of a portable document file (PDF), printer control language (PCL), and a PostScript file.

17. The system according to claim 14 further comprising:

a converter for converting the application file to a page description language file if the application file does not represent a page description language file.

18. The system according to claim 14 wherein the plug-in selector is adapted to access a plug-in database to retrieve the selected plug-in module.

19. The system according to claim 14 wherein the data augmenter cooperates with a downloader to express the preferential document-processing feature as downloader-embedded customization data in the application file.

20. The system according to claim 14 wherein the data augmenter cooperates with a printer driver to express the preferential document-processing feature as printer-driver-embedded customization data in the application file.

EXHIBIT B

figure 1 data processing system and printer

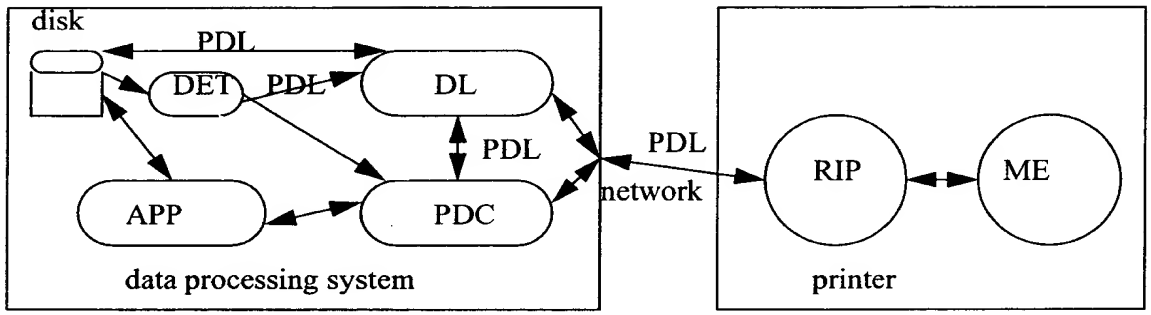
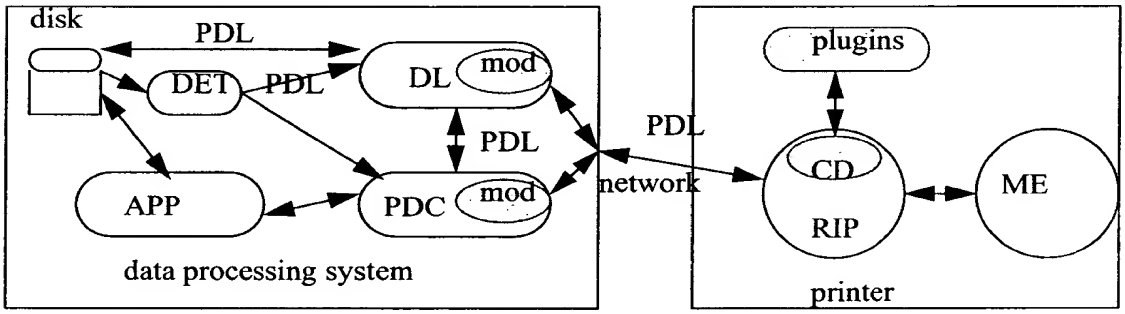


figure 2 data processing system and printer with customization



APP - application program e.g. Microsoft Word
CD - customization detector
DET- detector program to detect file types
DL- DownLoader - program which applys printing features
disk - disk drive
ME - Marking Engine - applys dots to paper, adds finishing features e.g. staple
mod - modifications done in program to support invention
PDC - printer driver converter - converts application files to PDL and applys printing features
PDL- page description language e.g. PostScript PCl PDF
plugins - plugin modules - apply features
RIP - raster image processor - converts PDL to dots for ME

EXHIBIT C

```
/*****************************************************************************
**
*
*              Copyright Heidelberg Digital L.L.C. 1999-2002
*                           ALL RIGHTS RESERVED
*
*****************************************************************************
*/

/*****************************************************************************
**
*
*   FILE NAME:       CheckCustom.c
*
*   SCCS Release:    @(#)CheckCustom.c      1.19      %w
*   Newest Delta:    [redacted]
*
*   FUNCTION LIST:   CheckCustom()
*                    DetermineDLs()
*                    CheckHost() -- ifdef'ed out
*
*   GENERAL DESCRIPTION:     Determine if the customlib should be called
*                            by in order check doc requirements which
*                            came from a %KDKCustom in the job, command
*                            line or environmental variables, and the
*                            default from the config utility. return TRUE
*                            if it should be called. Fill in the proper
*                            list of string and function arguments.
*                            the calling function responsibility to alloc
*                            space for the StringToCall pointer.
*
*   REVISION HISTORY:
*
*   DATE       AUTHOR            FUNCTIONS/DATA MODIFIED
*   ====       ======            =======================
*   [redacted]                   CheckCustom to own lib, 6.x interfaces
*                                strtok to strtok_r for MT safe
*                                pulled out [redacted]
*                                updated actons MAX_CUSTOM_FUNC was string
*                                5.x interfaces
*                                ifdef CheckHost since [redacted]
*                                          fixed bug in [redacted]
*   [redacted]
*   [redacted]                   Fix G_CANCEL_JOB message.
*                                Added CancelJob().
*                                Added parser for DL's and args.
*                                strcpy to strncpy
*                                moved [redacted] features to CheckHost()
*                                   this version has some differences from the
*   [redacted]
*   [redacted]                   print out "NULL" for custom arg + Ts_Doc_Complete
*                                original
*
*   ADD HISTORY TO TOP
*
*****************************************************************************
*/
```

```
/** INCLUDE FILES **/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dlfcn.h>
#include <ctype.h>
#include "common.h"
#include "tslib.h"
[redacted]
#include "ts_custom.h"

/** DEFINES **/

#ifdef RIPVERSION2
#define RIPVERSION '2'
#endif
#ifdef RIPVERSION3
#define RIPVERSION '3'
#endif
#ifdef RIPVERSION4
#define RIPVERSION '4'
#endif
#ifdef RIPVERSION5
#define RIPVERSION '5'
#endif
#ifdef RIPVERSION6
#define RIPVERSION '6'
#endif

/** TYPEDEFS **/

/** MACRO DEFINITIONS **/

/** EXTERN FUNCTION DECLARATIONS **/

/** EXTERN DATA DECLARATIONS **/

/** GLOBAL FUNCTION DECLARATIONS **/

/** GLOBAL DATA DEFINITIONS **/
static T_Bool JobCanceled;
static void *dl_handles[MAX_CUSTOM_LIBS]; /* ptr's for dynamic library handle
*/

/** LOCAL FUNCTION DECLARATIONS **/
[redacted]
    char        *StringArg,
    char        **FunctionsToUse,
    char        **StringsToUse,
    void        (**StartProcsToCall)(Doc_Requirements *, char *),
    int         (**ImageProcsToCall)(Ts_Image_Complete *, char *),
    void        (**EndProcsToCall)(    Doc_Requirements *,
                                       Ts_Doc_Complete *,
                                       char *) );

#ifdef SECURITY
static int CheckHost( void );
#endif

static void CancelJob( void );
```

[redacted]

```
/** LOCAL DATA DEFINITIONS **/
```

```
/*****************************************************************************
**
*
*    FUNCTION NAME:       CheckCustom
*
*    RETURN VALUE:        none
*
*    FORMAL ARGUMENTS
*                         pdr = document requirements
*                         CommandLine = TRUE if set via command line/enviroment
*                         CommandLineString = string if CommandLine TRUE
*                         FunctionsToUse = tells calling process function(s)
name
*                                          (NULL terminated list)
*                         StringsToUse = tells calling process string(s) to use
*                         StartProcsToCall = procedures to call at TS job start
*                         ImageProcsToCall = procedures to call for each image
*                         [redacted]
*
*    IMPLICIT INPUTS/OUTPUTS:
*
*    DESCRIPTION:
*
*    REVISION HISTORY:
*
*    DATE         AUTHOR      DESCRIPTION OF CHANGE
*    ====         ======      =====================
*    [redacted]   original
*
*****************************************************************************
*/
T_Bool
CheckCustom  (
[redacted]
    T_Bool                CommandLine,
    char                 *CommandLineString,
    char                 **FunctionsToUse,
    char                 **StringsToUse,
    void                 (**StartProcsToCall)(Doc_Requirements  *, char *),
    int                  (**ImageProcsToCall)(Ts_Image_Complete *, char *),
    void                 (**EndProcsToCall)  (Doc_Requirements  *,
                                              Ts_Doc_Complete *,
                                              char *) )
{
    static char __PROC__[] = "CheckCustom()";
    T_Bool      CallCustom;

    ts_printf(TS_DB_PROC)("%s %s - entry\n", __HEAD__, __PROC__);

    ts_printf(TS_DB_INFO)
        ("%s %s - KDK Act ON for Custom Lib = %s\n",
        __HEAD__, __PROC__,
            print_acton(pdr->kdk_acton_req.ka_Custom_lib));

[redacted]
        ("%s %s - Doc Requirements Use Custom Lib = %s\n",
        __HEAD__, __PROC__,
            /*CONSTCOND*/
        pdr->use_custom_lib == FALSE ? "FALSE" :
```

```
        pdr->use_custom_lib == TRUE ? "TRUE" :
        "UNKNOWN");

    if(pdr->custom_lib_arg == (char *)NULL)
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - Doc Requirements Custom string arg = NULL\n",
            __HEAD__, __PROC__);
    }
    else
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - Doc Requirements Custom string arg = %s\n",
            __HEAD__, __PROC__,
            pdr->custom_lib_arg);
    }

    ts_printf(TS_DB_INFO)
        ("%s %s - Command Line argument(-c) is %s\n",
        __HEAD__, __PROC__,
      /*CONSTCOND*/
        CommandLine == FALSE ? "FALSE" :
        CommandLine == TRUE ? "TRUE" :
        "UNKNOWN");

[redacted]
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - Command Line string argument = NULL\n",
[redacted]
        ;
    }
    else
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - Command Line string argument = %s\n",
            __HEAD__, __PROC__, CommandLineString);
        ;
    }

    /*  If ACT ON is set with a PL_TEMP, PL_USER_MODIFY PL_HEADER PL_PDL or
            PL_RESTORE_VALUE we will do what ever is requested by the
        doc requirements use_custom_lib field - either call the Custom lib
        or not. If Act ON is false we then check the -c command line option
        and if that is set we call the Custom lib. Finally, if both cases
        preceding are false we check the doc requirements use_custom_lib
        field. This reflects the system default.
    */

    if(pdr->kdk_acton_req.ka_Custom_lib)
    {
            /*CONSTCOND*/
        if(pdr->use_custom_lib == TRUE)
        {
            ts_printf(TS_DB_INFO)
                ("%s %s - Calling Custom() due to %KDKCustom:\n",
            __HEAD__, __PROC__);

#ifdef SECURITY
[redacted]
            {
```

```
                        return(0);
            }
#endif

                    /*CONSTCOND*/
[redacted]
            DetermineDLs(   pdr->custom_lib_arg,
                            FunctionsToUse,
                            StringsToUse,
                            StartProcsToCall,
                            ImageProcsToCall,
                            EndProcsToCall);
        }

        else
        {
            ts_printf(TS_DB_INFO)
                ("%s %s - *NOT* Calling Custom() due to %%KDKCustom:\n",
                __HEAD__, __PROC__);
            ;
            CallCustom = FALSE;
        }

    }

      /*CONSTCOND*/
    else if (CommandLine == TRUE)
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - Calling Custom() due to command line arg\n",
            __HEAD__, __PROC__);

#ifdef SECURITY
[redacted]
        {
                return(0);
        }
#endif

            /*CONSTCOND*/
[redacted]
        DetermineDLs(   CommandLineString,
                        FunctionsToUse,
                        StringsToUse,
                        StartProcsToCall,
                        ImageProcsToCall,
                        EndProcsToCall);
    }

        /*  if act on is off but this is true it was set via the
            config utility  */
      /*CONSTCOND*/
    else if(pdr->use_custom_lib == TRUE)
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - Calling Custom() due to sys default:\n",
            __HEAD__, __PROC__);

#ifdef SECURITY
        if(CheckHost() == 0)
        {
```

```
                    return(0);
        }
#endif

            /*CONSTCOND*/
        CallCustom = TRUE;

[redacted]
                            FunctionsToUse,
                            StringsToUse,
                            StartProcsToCall,
                            ImageProcsToCall,
                            EndProcsToCall);
    }

    else
    {
[redacted]
        ("%s %s - *NOT* Calling Custom()\n", __HEAD__, __PROC__);
        CallCustom = FALSE;
    }

    ts_printf(TS_DB_PROC)("%s %s - exit\n", __HEAD__, __PROC__);

    return(CallCustom);
}
```

```
/*****************************************************************************
**
*
*    FUNCTION NAME:       DetermineDLs
*
*    RETURN VALUE:        none
*
*    FORMAL ARGUMENTS     StringArg = string to parse
*                         FunctionsToUse = tells calling process function(s)
name
*                                         of the ImageProcsToCall. SInce it is
*                                         a NULL terminated list it also lets
you
*                                         know how many calls to make
*                         StringsToUse = tells calling process string(s) to use
*                                         can be NULL in middle of list
[redacted]
*                                         can't be NULL in middle of list
*                         ImageProcsToCall = function(s) to call for each image
*                                         can't be NULL in middle of list
*                         EndProcsToCall = function(s) to call at job end
*                                         can't be NULL in middle of list
*
*    IMPLICIT INPUTS/OUTPUTS:
*
*    DESCRIPTION:         Extract DLs from KDKCustom: string if present.
*
*    REVISION HISTORY:
*
*    DATE          AUTHOR       DESCRIPTION OF CHANGE
*    ====          ======       =====================
*    [redacted]    Added dl-parser.
*    [redacted]    original
*
*****************************************************************************
*/
static void
DetermineDLs(
                char     *StringArg,
                char    **FunctionsToUse,
                char    **StringsToUse,
                void      (**StartProcsToCall)(Doc_Requirements *, char *),
                int       (**ImageProcsToCall)(Ts_Image_Complete *, char *),
                void      (**EndProcsToCall)  (Doc_Requirements *,
                                               Ts_Doc_Complete  *,
                                               char *)
            )
{

    int i = 0;                      /* generic counter */
    char *startptr; /* start point for parsing of string */

[redacted]
    char *subtokptr;

    char *tmpptr;
    char *tmpptr2;

    char *dl_args[MAX_CUSTOM_LIBS]; /* DL'-string-args array */
```

```
char  buf[MAX_CUSTOM_STRING];

T_Bool USE_DLL = FALSE;

char bc[128]; /* back channel error strings */

static char __PROC__[] = "DetermineDLs()";

JobCanceled = FALSE;

ts_printf(TS_DB_PROC)("%s %s - entry\n", __HEAD__, __PROC__);

    /*  if no -d function is Custom | CustomEndOfJob | CustomStartOfJob
        It is Custom vs. CustomImage for backwards compatibility
        if format is -d e.g. "-dStore:"arg 1";Mail;CleanUp:arg2"
        first function is KDKCustomStoreStartOfJob |
        KDKCustomStoreEndOfJob | KDKCustomStoreImage and the
        first string is "arg 1" - the 2nd function is
        KDKCustomMailStartOfJob etc with a NULL string and 3rd
        KDKCustomCleanUpStartOfJob with "arg2" as the string.
        An alias file will allow the site to remap oprions
        such as alias s store or alias s store:helloworld
    */
strncpy(StringsToUse[0], StringArg, MAX_CUSTOM_STRING);

if(StringsToUse[0] == (char *)NULL)
{
    ts_printf(TS_DB_INFO)
[redacted]
}
else
{
    ts_printf(TS_DB_INFO)
        ("%s\tFunction-Arg string = (%s)\n", __HEAD__, StringsToUse[0]);
}

i = 0;
while ( i < MAX_CUSTOM_LIBS )  /* Init Procs */
{
[redacted]
    ImageProcsToCall[i] = DEFAULT_IMAGE;
    EndProcsToCall[i]   = DEFAULT_ENDOFJOB;
    i++;
}

/* init 1st for DEFAULT use of custom */
strncpy(FunctionsToUse[0], STRING_IMAGE, MAX_CUSTOM_FUNC);

if ( (startptr = strstr(StringArg, "-d")) != (char *) NULL )
  {
        /*CONSTCOND*/
    USE_DLL = TRUE;
    ts_printf(TS_DB_INFO)("%s %s - -d Dynamic mode found! Using DLs.\n",
                __HEAD__,__PROC__);
}
else
{
    i = 1;
    while ( i < MAX_CUSTOM_LIBS ) /* skip the remaining functions */
    {
        *FunctionsToUse[i] = '\0';
```

```
            i++;
        }
    }

        ("%s\tDEFAULT: Using Function = (%s)\n", __HEAD__,
FunctionsToUse[0]);

    ts_printf(TS_DB_INFO)("%s\tString-Arg = (%s)\n",
          __HEAD__, StringArg);

    /*
     * Parse KDKcustom: (string)
     */

    /* Begin if dynamic lib mode */

    {
            char *lasts;

        startptr += strlen("-d"); /* bump up past '-d' */

            /************************************
             *      Parse KDKCustome string      *
             ************************************/

        /*
         * Break up input string by: ';'
         * "dl1:"args1 args1b";dl2:args2;..."
         * strings delim by ';'
         */
        i = 0;
        while( ((tokptr = strtok_r(startptr, ";", &lasts )) != NULL) && i <
MAX_CUSTOM_LIBS ) /* clip off -d */
        {
            dl_args[i] = tokptr;

            ts_printf(TS_DB_INFO)("%s\tdl-args[%d] = (%s)\n",
                    __HEAD__, i, tokptr);

            i++;

        }


        while ( i < MAX_CUSTOM_LIBS ) /* NULL out the rest of dl_args array
if no more functions */
        {
            i++;
        }

        /*
         *  Break up "dl:args" string by ":" to get "dl-args".
         */

        i = 0;
        while( ((tokptr = strtok(dl_args[i], ":")) != NULL)
                && i < MAX_CUSTOM_LIBS
                && dl_args[i] != (char *) NULL)
```

```
        {
                tmpptr  = FunctionsToUse[i];
                tmpptr2 = tokptr;

            /*
             * Strip off white-space
             */
            while( !(*tmpptr2 == '\0') )
            {
                if( !isspace(*tmpptr2) )
                *(FunctionsToUse[i]++) = *tmpptr2;
                tmpptr2++;
            }
            *(FunctionsToUse[i]) = '\0';
[redacted]


            ts_printf(TS_DB_INFO)("%s\tFunctionsToUse[%d] = (%s)\n",
                    __HEAD__, i, FunctionsToUse[i]);


            subtokptr = strtok((char *)NULL, ":"); /* get other half of
string */

[redacted]
                __HEAD__, i, subtokptr?subtokptr:"NULL");

            if ( subtokptr != (char *) NULL )
            {

                if ( strlen(subtokptr) == 0)
                {
                    *StringsToUse[i] = '\0';
                }
                else
                {
                    strncpy( StringsToUse[i], subtokptr, MAX_CUSTOM_STRING );
                }
            }
            else
                *StringsToUse[i] = '\0';

            i++;
        }


        while ( i < MAX_CUSTOM_LIBS ) /* NULL out the rest of array if no
more functions */
        {
[redacted]
            i++;
        }

        /*
         *  Check and open dl's.
         */
        i = 0;
        while ( (FunctionsToUse[i] != (char *)NULL) &&
                (strlen(FunctionsToUse[i])  != 0)   &&
[redacted]
```

```
{
    sprintf(buf, "/hp/lib/KDKCustom%s.so", FunctionsToUse[i] );

    if ((dl_handles[i] = dlopen(buf, RTLD_LAZY)) == NULL)
    {

        sprintf(bc, "dlopen: %s\n", dlerror() );

        ts_printf(TS_DB_ERRORS)("%s %s - %s\n",
            __HEAD__, __PROC__, bc);

        LogBackChannel (SaveQueue((long)0), bc, strlen(bc));

        *FunctionsToUse[i] = '\0';

        CancelJob();

        break;

    }
    else
        ts_printf(TS_DB_INFO)
                ("%s\tFound DLL(%s)\n", __HEAD__, FunctionsToUse[i]);


/*
[redacted]
 * and assign them to Procs.
 *
 * Note: Use program "cdecl" to read cast
 *
 */

    if ( (StartProcsToCall[i] =
[redacted]
"CustomStartOfJob")) == NULL )
    {
        sprintf(bc, "dlsym: %s\n", dlerror());

        ts_printf(TS_DB_ERRORS)("%s %s -\n\t%s\n",
            __HEAD__, __PROC__, bc );


        LogBackChannel(SaveQueue((long)0), bc, strlen(bc));

        StartProcsToCall[i] = '\0';

        CancelJob();

        break;
    }
    else
        ts_printf(TS_DB_INFO)("%s\tFound (%s):CustomStartOfJob()\n",
                __HEAD__, FunctionsToUse[i]);


    if ( (ImageProcsToCall[i] =
        (int (*)(Ts_Image_Complete *, char *)) dlsym(dl_handles[i],
                "Custom")) == NULL )
    {
[redacted]
```

```
                ts_printf(TS_DB_ERRORS)("%s %s -\n\t%s\n",
                    __HEAD__, __PROC__, bc );

                LogBackChannel(SaveQueue((long)0), bc, strlen(bc));

                ImageProcsToCall[i] = '\0';

[redacted]

                break;

            }
            else
                ts_printf(TS_DB_INFO)
                        ("%s\tFound (%s):Custom()\n", __HEAD__,
FunctionsToUse[i]);


            if ( (EndProcsToCall[i] =
                (void (*)(Doc_Requirements *,
                        Ts_Doc_Complete *, char *)) dlsym(dl_handles[i],
                        "CustomEndOfJob")) == NULL )
            {
                sprintf(bc, "dlsym: %s\n", dlerror());

                ts_printf(TS_DB_ERRORS)("%s %s -\n\t%s\n",
                    __HEAD__, __PROC__, bc );

                LogBackChannel (SaveQueue((long)0), bc, strlen(bc));

                EndProcsToCall[i] = '\0';

                CancelJob();

[redacted]

            }
            else
                ts_printf(TS_DB_INFO)("%s\tFound (%s):CustomEndOfJob()\n",
__HEAD__, FunctionsToUse[i]);

            i++;

        } /* End of while FunctionsToUse[i] */

    } /* End if dynamic lib mode */

[redacted]
}
```

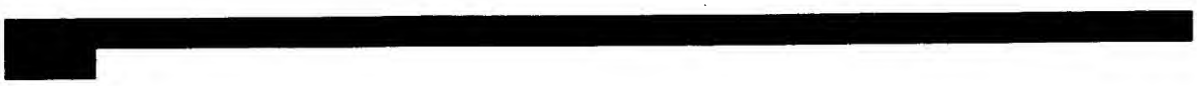
```
/*****************************************************************************
**
*
*   FUNCTION NAME:      CheckHost
*
*   RETURN VALUE:       none
*
*   FORMAL ARGUMENTS
*
*   IMPLICIT INPUTS/OUTPUTS:
*
*   DESCRIPTION:
*
*   REVISION HISTORY:
*
*   DATE        AUTHOR      DESCRIPTION OF CHANGE
*   ====        ======      =====================
*   [redacted]  original
*
*****************************************************************************
*/
#ifdef SECURITY
static int
CheckHost( void )
{
    static char __PROC__[] = "CheckHost()";
    static char     command[] = "/usr/bin/pkginfo -l KDKrip | /usr/bin/grep
VERSION | awk '{ print $2 }'";
    unsigned long   hostid;
    unsigned long   validhostid;
#if 1 /* swich back if if 0 */
    static char     hoststr[] = "0X00EDEDED";
#else
      static char hoststr[] = "0X80a6b329";      /*     EdC          */
[redacted]
      static char hoststr[] = "0X808cf598";      /*     dhvem21      */
#endif

    FILE            *the_pipe;
    unsigned char   buffer[ARG_SIZE];

    ts_printf(TS_DB_PROC)("%s %s - entry\n", __HEAD__, __PROC__);

    hostid = (unsigned long)gethostid();

    ts_printf(TS_DB_INFO) ("%s %s -     validating hostid = 0X%x\n",
        __HEAD__, __PROC__, hostid);

[redacted]

    if (hostid != validhostid)
    {
        char    bc[128];

        sprintf (bc, "%s Invalid hostid expected: 0x%x\n", VERSION,
validhostid);

        ts_printf(TS_DB_INFO) ("%s %s - %s\n", __HEAD__, __PROC__, bc);

        LogBackChannel (SaveQueue((long)0), bc, strlen(bc));
```

```
        /*  this will be unknown error until 03.00  */
    ts_interp_error(    (int)SaveQueue((long)0),
                        TS_WARN_HOST_ID,
                        __PROC__,
                        (char *)NULL,
                        FALSE);

    return(0);
}

    /*   now check if RIP version number is ok  */

ts_printf(TS_DB_INFO) ("%s %s -     validating RIP version\n",
    __HEAD__, __PROC__);

memset(buffer, '\0', ARG_SIZE);

the_pipe = popen(command,"r");

if(fread(buffer, 1, ARG_SIZE, the_pipe) == 0)
{
    char    bc[128];

    ts_printf(TS_DB_INFO) ("%s %s - %s\n", __HEAD__, __PROC__, bc);

    LogBackChannel (SaveQueue((long)0), bc, strlen(bc));

        /*  this will be unknown error until 03.00  */
    ts_interp_error(    (int)SaveQueue((long)0),
                        TS_WARN_POPEN,
                        __PROC__,
                        (char *)NULL,
                        FALSE);

    ts_printf(TS_DB_PROC) ("%s %s - exit\n", __HEAD__, __PROC__);
    return(0);
}

ts_printf(TS_DB_INFO) ("%s %s - RIP version = %s\n",
    __HEAD__, __PROC__, buffer);

    /* this means works with any 03.xx.xx.xxxx version  */
if((buffer[0] != '0') || (buffer[1] != RIPVERSION))
{
    char    bc[128];

    sprintf (bc, "%s Invalid RIP version\n", VERSION);

    ts_printf(TS_DB_INFO) ("%s %s - %s\n", __HEAD__, __PROC__, bc);

    LogBackChannel (SaveQueue((long)0), bc, strlen(bc));

        /*  this will be unknown error until 03.00  */
        ts_interp_error( (int)SaveQueue((long)0),
```

```
                                __PROC__,
                                (char *)NULL,
                                FALSE );

        ts_printf(TS_DB_PROC) ("%s %s - exit\n", __HEAD__, __PROC__);
        return(0);
    }

    return(1);
}
#endif      /* SECURITY */
```

```
/*****************************************************************************
**
*
*   FUNCTION NAME:       CancelJob()
*
*   RETURN VALUE:        none
*
*   FORMAL ARGUMENTS
*
*   IMPLICIT INPUTS/OUTPUTS:
*
*   DESCRIPTION: Will cancel current job.
*
*****************************************************************************
*/
static void
CancelJob(void)
{
    Uint32 jobid;
[redacted]

    ts_printf(TS_DB_PROC) ("%s %s - entry\n", __HEAD__, __PROC__);

    if(JobCanceled)
    {
        ts_printf(TS_DB_PROC) ("%s %s - (job previously canceled) exit\n",
__HEAD__, __PROC__);
        return;
    }
      else
           /*CONSTCOND*/
        JobCanceled = TRUE;


    jobid = GetJobID();


    pgcj = (G_Cancel_Job *)valloc(sizeof(G_Cancel_Job));
[redacted]
    {
        ts_printf(TS_DB_PROC) ("%s %s - error malloc()\n", __HEAD__,
__PROC__);

        ts_interp_error( (int)SaveQueue((long)0),
                         TS_ERR_MALLOC,
                         __PROC__,
                         (char *)NULL,
                                   /*CONSTCOND*/
                         TRUE );
    }


    pgcj->ack        = (int)0;
    pgcj->response   = (int)0;
    pgcj->operation  = (int)0;
    pgcj->job_id     = (Uint32)jobid; /* Cancel last job */
    pgcj->connect_id = (long)0;
```

```
    ts_printf(TS_DB_PROC) ("%s %s - Canceling Job-%d\n",
                __HEAD__, __PROC__, jobid);
```



```
    free( (G_Cancel_Job *)pgcj );
    pgcj = (G_Cancel_Job *)NULL;

    ts_printf(TS_DB_PROC) ("%s %s - exit\n", __HEAD__, __PROC__);

    return;
}
```

```
/*****************************************************************************
**
*
*    FUNCTION NAME:       CloseCustom()
*
*    RETURN VALUE:        none
*
*    FORMAL ARGUMENTS
*
*    IMPLICIT INPUTS/OUTPUTS:
*
*    DESCRIPTION: Close Custom dll handles.
*
*****************************************************************************
*/
void
CloseCustom(void)
{
    char bc[128]; /* back channel error strings */
    int i;
    static char __PROC__[] = "CloseCustom()";

    ts_printf(TS_DB_PROC) ("%s %s - entry\n", __HEAD__, __PROC__);

    i = 0;
    while( (dl_handles[i] != (void *)NULL) &&  i < MAX_CUSTOM_LIBS )
    {
        ts_printf(TS_DB_ERRORS)("%s\t Closing handle %d\n",
                __HEAD__,i);

        if( dlclose(dl_handles[i]) != 0 )
        {
[redacted]

            ts_printf(TS_DB_ERRORS)("%s %s - %s\n",
                __HEAD__, __PROC__, bc);

            LogBackChannel(SaveQueue((long)0), bc, strlen(bc));

            CancelJob();

        }
[redacted]
        i++;
    }

    ts_printf(TS_DB_PROC) ("%s %s - exit\n", __HEAD__, __PROC__);

    return;
}
```

EXHIBIT D

```
/*****************************************************************************
*
*                                COPYRIGHT HEIDELBERG DIGITAL L.L.C. 1999-2000
*                                          ALL RIGHTS RESERVED
*
*****************************************************************************/

/*****************************************************************************
*
*      FILE NAME:          CustomWriteTiff.c
*
*      SCCS Release:       @(#)CustomWriteTiff.c   1.10
*      Newest Delta:       [redacted]
*
*      FUNCTION LIST:      Custom()
*
*      GENERAL DESCRIPTION:     This is the first custom implementation
*                                          This gives the ability to store tiff
files.
*
*      REVISION HISTORY:
*
*      DATE          AUTHOR              FUNCTIONS/DATA MODIFIED
*      ====          ======              =======================
*      [redacted]    [redacted]          5.x interfaces
*      [redacted]    [redacted]          HeaderFile and Title added
*      [redacted]    [redacted]          removed ripversion 02.xx defines
*      [redacted]    [redacted]          moved security features to CheckCustom()
*      [redacted]    [redacted]          3.0.x API
*      [redacted]    [redacted]          call script if it exists, RIP version defines
*      [redacted]    [redacted]          security features hostid and rip version check
*      [redacted]    [redacted]          original
*
*            ADD HISTORY TO TOP
*
*****************************************************************************/

/** INCLUDE FILES **/
#include <stdio.h>
#include <strings.h>
#include <stdlib.h>
#include <sys/shm.h>
#include <unistd.h>
#include <sys/types.h>
#include <fcntl.h>
#include "jm_ts.h"
[redacted]
#include "ts_custom.h"
#include "tslib.h"
#include "ts_job.h"
#include "RIP2Store.h"

/** DEFINES **/

/** TYPEDEFS **/
```

```
/** MACRO DEFINITIONS **/

/** EXTERN FUNCTION DECLARATIONS **/

/** EXTERN DATA DECLARATIONS **/

/** GLOBAL FUNCTION DECLARATIONS **/

/** GLOBAL DATA DEFINITIONS **/

/** LOCAL FUNCTION DECLARATIONS **/

/** LOCAL DATA DEFINITIONS **/

/*****************************************************************************
*
*     FUNCTION NAME:          Custom()
*
*     RETURN VALUE:           none
*
*     FORMAL ARGUMENTS: pointer to a Ts_Image_Complete struct
*                             args - the same args passed in in the
KDKCustom: "args"
*
*     IMPLICIT INPUTS/OUTPUTS:
*
*     DESCRIPTION:            if using this library in releases prior to
02.06.xx you
*                             would need to define SaveQueue (as 70) which
is in libts and
*                             recompile. Also there would not be text for
some error messages
*                             in the SM log. This may be done by defining
USE_WITH_OLD_VER
*
*     REVISION HISTORY:
*
*     DATE        AUTHOR            FUNCTIONS/DATA MODIFIED
*     ====        ======            =======================
*     [redacted]  original
*
*           ADD HISTORY TO TOP
*
*****************************************************************************/
int         Custom      (     Ts_Image_Complete *ptic,
                              char                    *args)
{
      static char       __PROC__[] = "Custom()";
      PSMBandData       band;
      static int        CallScript;
      static int        PrintRaster;
      int                     fd;
      char              *shmaddr;
      [redacted]
      char              scriptfile[128];
      static char       HeaderFile[MAX_RESOURCE_FILE_NAME_SIZE];
```

```
	static char		t[MAX_LABEL];
	static char		*Title;
	Doc_Requirements *dreqs;

	ts_printf(TS_DB_PROC) ("%s %s - entry\n", __HEAD__, __PROC__);

	Title = t;

	ts_printf(TS_DB_INFO) ("%s %s -	job_id = %d\n",
		__HEAD__, __PROC__, ptic->job_id);

	ts_printf(TS_DB_INFO) ("%s %s -	PageNumber = %d\n",
		__HEAD__, __PROC__, ptic->PageNumber);

	if(args == (char *)NULL)
	{
		ts_printf(TS_DB_INFO) ("%s %s -	arguments = NULL\n",
			__HEAD__, __PROC__);
		;
	}
	else
	{
		ts_printf(TS_DB_INFO) ("%s %s -	arguments = %s\n",
[redacted]
		;
	}

	ts_printf(TS_DB_INFO) ("%s %s -	Image pixels/line = %d\n",
		__HEAD__, __PROC__,
		ptic->image_size.image_pixels_line);

	ts_printf(TS_DB_INFO) ("%s %s -	Image lines page = %d\n",
		__HEAD__, __PROC__,
		ptic->image_size.image_lines);

		/* even if print range was like 3 6 the first page is still 1	*/
	if(ptic->PageNumber < 2)
	{
		dreqs = ts_get_doc_reqs(ptic->job_id);

		strncpy(HeaderFile, dreqs->header_file_name,
MAX_RESOURCE_FILE_NAME_SIZE);

			/* the file that will be unlinked when job is complete	*/
		ts_printf(TS_DB_INFO) ("%s %s -	HeaderFile is: = %s\n",
			__HEAD__, __PROC__, HeaderFile);

		if((dreqs->DSC_Title == (char *)NULL) || (dreqs->DSC_Title[0] == 0))
		{
			strcpy(Title, DEFAULT_TITLE);

			ts_printf(TS_DB_INFO) ("%s %s -	Title is: = NULL using
default %s\n",
				__HEAD__, __PROC__, DEFAULT_TITLE);
		}

		else
```

```
        {
              [redacted]

              ts_printf(TS_DB_INFO) ("%s %s -      Title is: = %s\n",
                    __HEAD__, __PROC__, Title);
        }

        if(strcmp(args,"NoPrint") == 0)
        {
              ts_printf(TS_DB_INFO)
                    ("%s %s -      Holding Print due to NoPrint string
arg\n",
                    __HEAD__, __PROC__);

                    PrintRaster = 0;
        }

        else
        {
              ts_printf(TS_DB_INFO)
                    ("%s %s -      **NOT Holding Print due to NoPrint string
arg\n",
                    __HEAD__, __PROC__);

                    PrintRaster = 1;
        }

        strcpy(scriptfile, CUSTOM_SCRIPT_DIR);
        strcat(scriptfile, CUSTOM_IMAGE_SCRIPT);
        if (fd = open(scriptfile, O_RDONLY) == -1)
        {
              ts_printf(TS_DB_INFO) ("%s %s - Will Not Call Script: %s\n",
                    __HEAD__, __PROC__, scriptfile);

              CallScript = 0;
        }

        else
        {
              close(fd);
              ts_printf(TS_DB_INFO) ("%s %s - Will Call Script: %s\n",
                    __HEAD__, __PROC__, scriptfile);

              CallScript = 1;
        }
  }

  [redacted]
  {
        ts_printf(TS_DB_INFO) ("%s %s -      No shmem ID ---\n",
              __HEAD__, __PROC__);

        ts_interp_error(  (int)SaveQueue((long)0),
                                    (u_short)TS_ERR_SHMID,
                                    (char *)__PROC__,
                                    (char *)NULL,
                                    (char)TRUE);
```

```
            return(PrintRaster);
      }

      shmaddr = (void *)0;

              /*  attach to get vitrual address */
      if ((shmaddr = (char *)shmat(ptic->shm_id, (void *)shmaddr, 0))
            == (void *)-1)
      {
            perror("shmat failed ...\n");
            ts_printf(TS_DB_INFO) ("%s %s -     attach to shared memory failed -
--\n",
                  __HEAD__, __PROC__);
            ts_interp_error(  (int)SaveQueue((long)0),
                                    (u_short)TS_ERR_SHMAT,
                                    (char *)__PROC__,
                                    (char *)NULL,
                                    (char)TRUE);
            exit(1);
      }
            /*    get a copy of the start of shared memory for the linked list
      */
      shmll = shmaddr;

            /*    go to the correct place in shared memory for this buffer
number      */
      shmll += ptic->LinkedListOffset;

#if 0
      [redacted]
      ts_printf(TS_DB_INFO)("ptic->LinkedListOffset = 0x%x  ", ptic-
>LinkedListOffset);
      ts_printf(TS_DB_INFO)("shmll = 0x%x\n", shmll);
#endif

            /*    this is where the linked list starts      */
      band = (PSMBandData)shmll;

            /*    *note - only 1 band in a banded frame model could be used
                  in order to reuse the same shared memory size. If camelot
                  wants to use more bands, it should become a rip.pids option
                  or it can stick with contiguous bands and no emptys */

      if (!band->offset)
      {
            ts_printf(TS_DB_INFO) ("%s %s -     Band EMPTY not supported ---\n",
                  __HEAD__, __PROC__);

            ts_interp_error(  (int)SaveQueue((long)0),
                                    TS_WARN_EMPTY_BAND,
                                    __PROC__,
                                    (char *)NULL,
                                    FALSE);

            shmdt((void *) shmaddr);
            return(PrintRaster);
```

```
        }

        if(band->lastBand)
        {
              WriteTiff(   (char *)(band->offset + shmaddr),
                           ptic->job_id,
                           ptic->PageNumber,
                           [redacted]
                           ptic->image_size.image_lines,
                           CallScript,
                           HeaderFile,
                           Title);
        }

        else
        {
              [redacted]
\n",
                    __HEAD__, __PROC__);

              ts_interp_error(   (int)SaveQueue((long)0),
                                 TS_WARN_MULT_BAND,
                                 __PROC__,
                                 (char *)NULL,
                                 FALSE);

              shmdt((void *) shmaddr);
              return(PrintRaster);
        }

        shmdt((void *) shmaddr);
        ts_printf(TS_DB_PROC) ("%s %s - exit\n", __HEAD__, __PROC__);

        return(PrintRaster);

}
#ifdef USE_WITH_OLD_VER
long SaveQueue(long queue)
{
    static long savequeue = 70;

    if (queue)
        savequeue = queue;

    return(savequeue);
}
#endif

void CustomStartOfJob   (     Doc_Requirements  *dreqs,
                              char                        *args)

{
      static char __PROC__[] = "CustomStartOfJob()";

      ts_printf(TS_DB_PROC)("%s %s -      ENTRY - EXIT\n",
            [redacted]
}
```

```
void CustomEndOfJob     (     Doc_Requirements  *dreqs,
                              Ts_Doc_Complete         *dcom,
                              char                    *args)
{
      static char __PROC__[] = "CustomEndOfJob()";

    ts_printf(TS_DB_PROC)("%s %s -     ENTRY - EXIT\n",
        __HEAD__, __PROC__);
}
```

EXHIBIT E

```
/*****************************************************************************
*
*                              Copyright Heidelberg Digital L.L.C. 1999-2002
*                                        ALL RIGHTS RESERVED
*
*****************************************************************************/

/*****************************************************************************
*
*      FILE NAME:          WriteTiff()
*
*      SCCS Release:       @(#)WriteTiff.c    1.11
*      Newest Delta:       [redacted]
*
*      FUNCTION LIST:      WriteTiff()
*
*      GENERAL DESCRIPTION:
*                                                  [redacted]
[redacted]
*      REVISION HISTORY:
*
*      DATE          AUTHOR              FUNCTIONS/DATA MODIFIED
*      ====          ======              =======================
*      [redacted]    6.x interfaces
*      [redacted]    5.x interfaces
*      [redacted]    Change TIFF orintation tag in compress lib
*      [redacted]    HeaderFile and Title added as new args for script
*      [redacted]    Added CallScript
*      [redacted]      moved version to header file
*      [redacted]    open file and change orientation to 8
*      [redacted]    changed filename from Tiff.xxxxx to xxxxx.tif
*      [redacted]    original
*
*            ADD HISTORY TO TOP
*
*****************************************************************************/

/** INCLUDE FILES **/
#include <stdio.h>
#include <strings.h>
#include <sys/stat.h>
#include <sys/param.h>
[redacted]
#include <sys/types.h>
#include <unistd.h>
#include "stdtype.h"
#include "jm_ts.h"
#include "ts_custom.h"
#include "tslib.h"
#include "RIP2Store.h"

      /* [redacted]
[redacted]
      */
```

```
#if 0
#include <thread.h>
static void    script_thread(int NotUsed);
long  GJobID;
int         GPage;
char  *GHeaderFile;
#endif

/** DEFINES **/

/** TYPEDEFS **/

/** MACRO DEFINITIONS **/

/** EXTERN FUNCTION DECLARATIONS **/
extern void       ChangeTiffOrientationTag(int);

/** EXTERN DATA DECLARATIONS **/

/** GLOBAL FUNCTION DECLARATIONS **/

/** GLOBAL DATA DEFINITIONS **/

/** LOCAL FUNCTION DECLARATIONS **/
#if 0 /* needed in releases before 03.02.xx     */
static void ChangeTiffTag(char* filen, char *BC);
#endif

/** LOCAL DATA DEFINITIONS **/

/**   FUNCTIONS **/

/*****************************************************************************
*
*     FUNCTION NAME:    WriteTiff()
*
*     RETURN VALUE:     none
*
*     FORMAL ARGUMENTS:
*                           image - raster image to convert to TIFF
*                           JobID -  job number, used in dir name
*                           Page -  page number, used in filename
*                           width - width in pixels
*                           height - height in scan lines
*                           CallScript - call script with each page
*                           HeaderFile - name of file that will be unlinked at
job end
*                           Title - job name from %%Title
*
*     IMPLICIT INPUTS/OUTPUTS:
*
*     DESCRIPTION:
*
*     REVISION HISTORY:
*
*     DATE        AUTHOR      FUNCTIONS/DATA MODIFIED
*     ====        ======      =======================
```

```
*      [redacted]  original
*
*      ADD HISTORY TO TOP
*
*****************************************************************************/
void  WriteTiff(  char  *image,
                              long  JobID,
                              int         Page,
                              int         width,
                              int         height,
                              int         CallScript,
                              char  *HeaderFile,
                              char  *Title)
{
      static char       __PROC__[] = "WriteTiff()";
      static char       dname[MAXPATHLEN + 1];
      [redacted]
      char              bc[MAXPATHLEN + MAXNAMELEN + 100];
      u_long                  avail;
      int                     ras_length;
      int                     max_size;
      int                     pid;
      int                     i = 1;
      int                     once = 1;

      ts_printf(TS_DB_PROC) ("%s %s - entry queue = %d\n",
            __HEAD__, __PROC__, SaveQueue((long)0));

#if 0
GJobID = JobID;
GPage = Page;
GHeaderFile = HeaderFile;
#endif
            /*    only on page 1    */
      if(Page < 2)
      {
                  /*    make the default dir if it does not exist */
            if(mkdir((char *)DEFAULT_DIR, UMASK) < 0)
            {
                  if(errno != EEXIST)
                  {
                        ts_printf(TS_DB_ERRORS)
                              ("%s %s - Can't Create RIP2Store default
directory: %s\n",
                              __HEAD__, __PROC__, DEFAULT_DIR);

                        sprintf (bc, "%s Can't Create default directory: %s\n",
                              VERSION, DEFAULT_DIR);

                        LogBackChannel (SaveQueue((long)0), bc, strlen(bc));

                        ts_interp_error(  (int)SaveQueue((long)0),
                                                TS_WARN_DIR,
                                                __PROC__,
                                                [redacted]
                                                FALSE);
```

```
                        return;
                }
        }

                /*      now make the job dir, if it exists, keep trying by
appending a
                        number at the end (e.g. .1)    */

                /* the 5 if for 5 digits jobs 00000 - 99999      */
        sprintf (dname, "%s%.5d", DEFAULT_DIR, JobID);

        while (i != 0)
        {
                if(mkdir((char *)dname, UMASK) < 0)
                {
                        if(errno != EEXIST)
                        {
                                sprintf (bc, "%s Can't Create directory: %s\n",
VERSION, dname);
                                LogBackChannel (SaveQueue((long)0), bc,
strlen(bc));

                                ts_printf(TS_DB_ERRORS)
                                        ("%s %s - Can't Create RIP2Store directory:
%s\n",
                                        __HEAD__, __PROC__, dname);

                                ts_interp_error(   (int)SaveQueue((long)0),
                                                        TS_WARN_DIR,
                                                        __PROC__,
                                                        (char *)NULL,
                                                        FALSE);

                                return;
                        }

                        else
                        {
                                ts_printf(TS_DB_INFO)
                                        ("%s %s - Can't Create RIP2Store directory
(exists already): %s\n",
                                        __HEAD__, __PROC__, dname);

                                sprintf (bc, "%s Can't Create directory (exists
already): %s\n",
                                        VERSION, dname);
[redacted]

                                /* the 5 if for 5 digits jobs 00000 - 99999
        */
                                sprintf (dname, "%s%.5d.%d",DEFAULT_DIR, JobID,
i);

                                i++;
                        }
                }
```

```
                else
                {
                        ts_printf(TS_DB_INFO)("%s %s - Creating RIP2Store
directory: %s\n",
                                __HEAD__, __PROC__, dname);

                        sprintf (bc, "%s Creating directory: %s\n", VERSION,
dname);
                        LogBackChannel (SaveQueue((long)0), bc, strlen(bc));

                        i = 0;
                }
        }
    }

        /* the 5 if for 5 digits pages 00000 - 99999    */
    sprintf (fname, "%s/%.5d.tif",dname, Page);

    ras_length = (width * height) / 8;
    max_size = (int)( MAX_EXPAND * (float)ras_length / (float)1024);

    while ((avail = disk_kbytes_avail(dname)) < max_size)
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - waiting for disk space needed = %u kb  available =
%u kb\n",
            __HEAD__, __PROC__, max_size, avail);

        sleep(2);
        /* TBD time out   */

            /* report one time per image  */
        [redacted]
        {
            sprintf (bc, "%s waiting for disk space in directory: %s\n",
                VERSION, dname);

            LogBackChannel (SaveQueue((long)0), bc, strlen(bc));
            ts_interp_error(  (int)SaveQueue((long)0),
                                    TS_WARN_DISK_SPACE,
                                    __PROC__,
                                    (char *)NULL,
                                    FALSE);

            once = 0;
        }
    }

    ts_printf(TS_DB_INFO)("%s %s - disk space needed = %u kb available = %u
kb\n",
        __HEAD__, __PROC__, max_size, avail);

    ts_printf(TS_DB_INFO)("%s %s - Setting TIFF Orientation Flag to 8\n",
        __HEAD__, __PROC__);

    ChangeTiffOrientationTag(8);
```

```
      WriteTiffFile((long)10, fname, image, width, height, (int)0);

#if 0 /* needed in releases before 03.02.xx     */
      ChangeTiffTag(fname, bc);
#endif

      if(CallScript)
      {
            char P[128];
            char f[128];
            char j[8], p[8], w[8], h[8];

            strcpy  (P, CUSTOM_SCRIPT_DIR);
            strcat  (P, CUSTOM_IMAGE_SCRIPT);
            sprintf (f, "%.5d.tif", Page);
            sprintf (j, "%d", JobID);
            sprintf (p, "%d", Page);
            sprintf (w, "%d", width);
            sprintf (h, "%d", height);

            [redacted]

            ts_printf(TS_DB_INFO) (" %s %s %s %s %s\n",
                  p, w, h, HeaderFile, Title);

            if((pid = fork1()) == 0)      /*    child */
            {
                  execlp(P, CUSTOM_IMAGE_SCRIPT, dname, f, j, p, w, h,
                        HeaderFile, Title, (char *) 0);
                  _exit(1);
            }

            else
            {
                  ts_printf(TS_DB_INFO)("%s %s - Fork PID = %d\n",
                        __HEAD__, __PROC__, pid);
                  ;
            }

#if 0 /* Trying threaded version */
      {
      thread_t    tid;
      thr_create(     NULL,
                  0,
                  (void *(*)(void *))script_thread,
                  (void *)0,
                  THR_BOUND,
                  &tid);
      }
#endif

            [redacted]

            {
            char com[1024];
            strcpy  (com, CUSTOM_SCRIPT_DIR);
            strcat  (com, "a.out ");
```

```
            strcat  (com, f);
            strcat  (com, " ");
            strcat  (com, dname);
            strcat  (com, " ");
            strcat  (com, f);
            strcat  (com, " ");
            strcat  (com, j);
            strcat  (com, " ");
            strcat  (com, p);
            strcat  (com, " ");
            strcat  (com, w);
            strcat  (com, " ");
            strcat  (com, h);
            strcat  (com, " ");
            strcat  (com, HeaderFile);
            strcat  (com, " ");
            strcat  (com, Title);

            ts_printf(TS_DB_INFO) ("%s %s - Calling: %sa.out %s %s %s ",
                  __HEAD__, __PROC__, CUSTOM_SCRIPT_DIR, dname, f, j);

            ts_printf(TS_DB_INFO) (" %s %s %s %s %s\n",
                  p, w, h, HeaderFile, Title);

            system(com);
            }
#endif

      }

      ts_printf(TS_DB_PROC) ("%s %s - exit\n", __HEAD__, __PROC__);
}

#if 0 /* no longer used */
/*****************************************************************************
*
*     FUNCTION NAME:    ChangeTiffTag()
*
*     RETURN VALUE:     none
*
*     FORMAL ARGUMENTS:
*
*     IMPLICIT INPUTS/OUTPUTS:
*
*     DESCRIPTION:      Read through the tiff header and change the orientation
*                             from type 1 to type 8 (see line 0000160)
*
*           Before
*     0000000 4949 2a00 0800 0000 0f00 fe00 0400 0100
*     0000020 0000 0000 0000 0001 0400 0100 0000 e019
*     0000040 0000 0101 0400 0100 0000 0014 0000 0201
*     0000060 0300 0100 0000 0100 0000 0301 0300 0100
*     0000100 0000 0400 0000 0601 0300 0100 0000 0000
*     0000120 0000 0a01 0300 0100 0000 0100 0000 1101
*     0000140 0400 0100 0000 d200 0000 1201 0300 0100
*     0000160 0000 0100 0000 1501 0300 0100 0000 0100
*     0000200 0000 1601 0400 0100 0000 0014 0000 1701
```

```
*
*           After
*      0000000 4949 2a00 0800 0000 0f00 fe00 0400 0100
*      0000020 0000 0000 0000 0001 0400 0100 0000 e019
*      0000040 0000 0101 0400 0100 0000 0014 0000 0201
*      0000060 0300 0100 0000 0100 0000 0301 0300 0100
*      0000100 0000 0400 0000 0601 0300 0100 0000 0000
*      0000120 0000 0a01 0300 0100 0000 0100 0000 1101
*      0000140 0400 0100 0000 d200 0000 1201 0300 0100
*      0000160 0000 0800 0000 1501 0300 0100 0000 0100
*      0000200 0000 1601 0400 0100 0000 0014 0000 1701
*
*      Possible Tiff tags -    Default is 1.
*      1 =  The 0th  row represents the visual top of the image, and the
*           0th column represents the visual left hand side.
*      2 =  The 0th  row represents the visual top of the image, and the
*           0th column represents the visual right hand side.
*      3 =  The 0th  row represents  the visual bottom of the image, and
*           the 0th column represents the visual right hand side.
*      4 =  The 0th  row represents  the visual bottom of the image, and
*           the 0th column represents the visual left hand side.
*      5 =  The 0th  row represents  the visual  left hand  side of  the
*           image, and the 0th column represents the visual top.
*      6 =  The 0th  row represents  the visual  right hand  side of the
*           image, and the 0th column represents the visual top.
*      7 =  The 0th  row represents  the visual  right hand  side of the
*           image, and the 0th column represents the visual bottom.
*      8 =  The 0th  row represents  the visual  left hand  side of  the
*           image, and the 0th column represents the visual bottom.
*
*      REVISION HISTORY:
*
*      DATE         AUTHOR       FUNCTIONS/DATA MODIFIED
*      ====         ======       ======================
*      [redacted]                original
*
*      ADD HISTORY TO TOP
*
*****************************************************************************/
void  ChangeTiffTag(char *filen, char *BC)
{
      static char       __PROC__[] = "ChangeTiffTag()";
      char  junk[BUFFSIZE];
      int         fd;
      int         tmp32;
      int         OffSet;
      short tag;
      short tmp16;
      short ByteOrder;
      short Entries;
      short i;

      ts_printf(TS_DB_PROC)
            ("%s %s - entry queue = %d\n", __HEAD__, __PROC__,
SaveQueue((long)0));

      fd = open(filen, O_RDWR, 0644);
```

```
      if(fd < 0)
      {
            ts_printf(TS_DB_INFO)
                  ("%s %s - file open problem: %s\n",
                  __HEAD__, __PROC__, filen);

            sprintf (BC, "%s Can't Change TIFF Orientation Tag - file open
problem): %s\n",
                  VERSION, filen);
            LogBackChannel (SaveQueue((long)0), BC, strlen(BC));
            return;
      }

      read(fd, &ByteOrder, 2);        /*    byte order  */
      [redacted]
      {
            ts_printf(TS_DB_INFO)
                  ("%s %s - Byte Order = REVERSED\n",
                  __HEAD__, __PROC__);
      }

      else if(ByteOrder == T_STANDARD)
      {
            ts_printf(TS_DB_INFO)
                  ("%s %s - Byte Order = STANDARD\n",
                  __HEAD__, __PROC__);
      }

      else
      {
            ts_printf(TS_DB_INFO)
                  ("%s %s - non TIFF file: %s\n",
                  __HEAD__, __PROC__, filen);

            sprintf (BC, "%s Can't Change TIFF Orientation Tag - non TIFF file):
%s\n",
                  VERSION, filen);
            LogBackChannel (SaveQueue((long)0), BC, strlen(BC));
            close(fd);
            return;
      }

      read(fd, &tmp16, 2);     /*    002a or 2a00 reversed   */
      read(fd, &tmp32, 4);     /*    first IFD offset  */

      if(ByteOrder == T_REVERSED)
      {
            OffSet = (tmp32>>24) + ((tmp32>>8)&0xff00) + ((tmp32<<8)&0xff0000) +
(tmp32<<24);
      }
            else
      {
            OffSet = tmp32;
      }

      ts_printf(TS_DB_INFO)
            ("%s %s - IFD Offset = 0x%x\n",
```

```
        __HEAD__, __PROC__, OffSet);
```

```
    while(OffSet>0)
    {           /*    get to first IFD offset */
        if(OffSet > BUFFSIZE)
        {
            read(fd, junk, BUFFSIZE );
        }
        else
        {
            read(fd, junk, OffSet);
        }

        OffSet -= BUFFSIZE;
    }

    read(fd, &tmp16, 2);          /*    number of entries */

    if(ByteOrder == T_REVERSED)
        Entries = (tmp16>>8) + (tmp16<<8);
    else
        Entries = tmp16;

    ts_printf(TS_DB_INFO)
        ("%s %s - IFD Entries = %d\n",
        __HEAD__, __PROC__, Entries);

    for(i=0; i< Entries; i++)
    {
        read(fd, &tmp16, 2);    /*    tag id should be fixed              */
        read(fd, junk, 6);            /*    type + count should be fixed  */

        if(ByteOrder == T_REVERSED)
        {
            tag = (tmp16>>8) + (tmp16<<8);
        }
        else
        {
            tag = tmp16;
        }
```

```
            __HEAD__, __PROC__, tag);

        if(tag == 0x0112)
        {
            ts_printf(TS_DB_INFO)("%s %s - Changing Tiff Orientation Tag
to 8\n",
                __HEAD__, __PROC__);

            if(ByteOrder == T_REVERSED)
                tmp32 = 0x08000000;
            else
                tmp32 = 0x00000008;
```

```
                write(fd, &tmp32, 4);    /*     count */
        }

        else
        {
                read(fd, &tmp32, 4);     /*     count */
        }
    }
    close(fd);
}
#endif

#if 0
void    script_thread(int NotUsed)
{
    static char         __PROC__[] = "script_thread()";
        char P[128];
        char f[128];
        char j[8], p[8], w[8], h[8];
        char com[1024];
    static char         dname[MAXPATHLEN + 1];

    ts_printf(TS_DB_PROC) ("%s %s - entry queue = %d\n",
        __HEAD__, __PROC__, SaveQueue((long)0));

        strcpy  (P, CUSTOM_SCRIPT_DIR);
[redacted]
        sprintf (f, "%.5d.tif", GPage);
        sprintf (j, "%d", GJobID);
        sprintf (p, "%d", GPage);
            /* the 5 if for 5 digits jobs 00000 - 99999     */
        sprintf (dname, "%s%.5d", DEFAULT_DIR, GJobID);

        strcpy  (com, P);
        strcat  (com, " ");
        strcat  (com, dname);
        strcat  (com, " ");
        strcat  (com, f);
        strcat  (com, " ");
        strcat  (com, j);
        strcat  (com, " ");
        strcat  (com, p);
        strcat  (com, " ");
        strcat  (com, "3333");
        strcat  (com, " ");
        strcat  (com, "5555");
        strcat  (com, " ");
        strcat  (com, GHeaderFile);
        strcat  (com, " ");
        strcat  (com, "Title");

        ts_printf(TS_DB_INFO) ("%s %s - Calling: %s\n",
            __HEAD__, __PROC__, com);

        system(com);

    ts_printf(TS_DB_PROC) ("%s %s - exit\n",
```

```
            __HEAD__, __PROC__);

thr_exit((void *)0);
}
#endif
```

EXHIBIT F

```
/*****************************************************************************
*
*                              Copyright Heidelberg Digital L.L.C. 1999-2002
*                                           ALL RIGHTS RESERVED
*
*****************************************************************************/

/*****************************************************************************
*[redacted]
*     FILE NAME:         RIP2Store.h
*
*     SCCS Release:      @(#)RIP2Store.h   1.8
*     Newest Delta:      [redacted]
*
*     GENERAL DESCRIPTION:    RIP2Store header file
*
*     REVISION HISTORY:
*
*     DATE          AUTHOR              MODIFICATIONS
*     ====          ======              =======================
*     [redacted]    6.x interfaces
*     [redacted]    5.x interfaces
*     [redacted]    4.x version - updated arg size
*     [redacted]    changed WriteTiff prototype added DefaultTitle
*     [redacted]    added buff size
*     [redacted]    ability to call a script
*     [redacted]    original
*
*           ADD HISTORY TO TOP
*****************************************************************************/

/** DEFINES **/

    /*  max tiff size based on 4.3 which is worst case tiff expansion ratio */
#define MAX_EXPAND                  4.3
#define DEFAULT_DIR                 "/var/spool/CUSTOM_FILES/"
#define DEFAULT_TITLE          "DefaultTitle"

    /*  this is or'ed with process umask     */
#define UMASK                       0x1ff

#define T_STANDARD                  0x4d4d
#define T_REVERSED                  0x4949
#define BUFFSIZE               1024

#if 0
#define VERSION "@(#)Custom RIP2Store NoError 1.00 [redacted]"
#define VERSION "@(#)Custom RIP2Store 1.01 [redacted]"
#define VERSION "@(#)Custom RIP2Store 1.02 [redacted]"
#define VERSION "@(#)Custom RIP2Store 1.03 [redacted]"
#define VERSION "@(#)Custom RIP2Store (3.x testing) 1.04 [redacted]"
#define VERSION "@(#)Custom RIP2Store (03.01.xx testing) 1.04 [redacted]"
#define VERSION "@(#)Custom RIP2Store (03.02.xx testing) 1.06 [redacted]"
#define VERSION "@(#)Custom RIP2Store (04.02.xx) 1.07 [redacted]"
#define VERSION "@(#)Custom RIP2Store ([redacted]) 1.08 [redacted]"
```

```
#else
#define VERSION "@(#)Custom RIP2Store ([redacted]) 1.09 [redacted]"
#endif

#define ARG_SIZE              256

/** TYPEDEFS **/

/** MACRO DEFINITIONS **/

/** EXTERN FUNCTION DECLARATIONS **/
extern void WriteTiff(  char  *image,
                                    long  JobID,
                                    int         Page,
                                    int         width,
                                    int         height,
[redacted]
                                    char  *HeaderFile,
                                    char  *Title);

extern ulong disk_kbytes_avail( char *dir);

/** EXTERN DATA DECLARATIONS **/
extern  char            __HEAD__[];

/** GLOBAL FUNCTION DECLARATIONS **/

/** LOCAL FUNCTION DECLARATIONS **/
```